

UNITED STATES PATENT APPLICATION

FOR

MULTI-TOKEN SEAL AND UNSEAL

INVENTORS:  
DAVID GRAWROCK

PREPARED BY:

JEFFREY B. HUTER  
INTEL CORPORATION  
5000 W. CHANDLER BLVD., CH6-404  
CHANDLER, AZ 85226-3699

(480) 554-4198

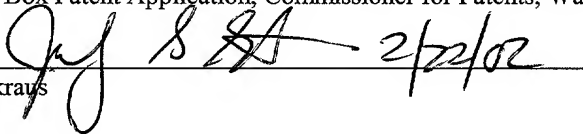
Attorney's Docket No. 42390.P13484

"Express Mail" mailing label number EL821777300US

Date of Deposit: February 22, 2002.

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Box Patent Application, Commissioner for Patents, Washington, D.C. 20231-0001

Judy L. Steinkraus

 2/22/02

# MULTI-TOKEN SEAL AND UNSEAL

## BACKGROUND

[0001] Existing software-based security services make the implicit assumption that a computing device or platform is trusted. They provide application-level security on the assumption that they execute in a safe environment. This assumption is true enough to justify the level of security required for existing business models, but state-of-the-art security functions are already providing the highest levels of protection that are possible without additional hardware support.

[0002] To this end, the Trusted Platform Computing Alliance (TPCA) describe in the TPCA Main Specification, Version 1.1a, 1 December 2001 a Trusted Platform Module (TPM) or physical token that provides increased confidence and that enables enhancements of existing services and new services. The TPM supports auditing and logging of software processes, platform boot integrity, file integrity, and software licensing. The TPM provides a protected store where items can be protected from exposure or improper use and provides an identity that can be used for attestation. These features encourage third parties to grant the platform access to information that would otherwise be denied.

[0003] The TPM contains an isolated computing engine whose processes can be trusted because they cannot be altered. These processes and the binding of the TPM to the platform can combine to reliably measure and report the state of the main computing environment inside the platform. The TPM provides a root of trust for the booting of the platform. While it is the Owner's responsibility to provide a safe operating system for a platform, once the OS has loaded, it can report the loading of untrusted software to the TPM before that untrusted software is loaded. The TPM can therefore report measured data that indicates the current state of the main software environment in the platform. A local or remote entity can simply query the TPM to reliably obtain these measurements and decide whether the platform's behavior enables it to be trusted for the intended purpose. Confidence in the loading of software is improved, because the TPM can attest to the current state of the operating system.

[0004] The TPM may act as a portal to confidential data, and may allow the release or use of that data only in the presence of a particular combination of access rights and software environment. Of course, the protected store of the TPM may be used for any sensitive data, not just identity information. The TPM may export these services to system-level software security services (such as IPSec) that are themselves called as services by ordinary applications. This arrangement enables greater confidence in the identity of a platform, while simultaneously allowing platform anonymity if so desired. Hence, any application that invokes proof of identity can be used with greater confidence and be allowed greater power.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The invention described herein is illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. For example, the dimensions of some elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference numerals have been repeated among the figures to indicate corresponding or analogous elements.

[0006] FIG. 1 illustrates an example computing device comprising a physical token and a virtual token.

[0007] FIG. 2 illustrates an example physical token and an example virtual token of FIG. 1.

[0008] FIG. 3 illustrates an example trusted operating environment that may be implemented by the computing device of FIG. 1.

[0009] FIG. 4 illustrates an example seal operation of the computing device of FIG. 1.

[0010] FIG. 5 illustrates an example unseal operation of the computing device of FIG. 1.

[0011] FIG. 6 illustrates an example multi-token seal operation of the computing device of FIG. 1.

[0012] FIG. 7 illustrates an example multi-token unseal operation of the computing device of FIG. 1.

[0013] FIG. 8 illustrates another example multi-token seal operation of the computing device of FIG. 1.

[0014] FIG. 9 illustrates another example multi-token unseal operation of the computing device of FIG. 1.

## DETAILED DESCRIPTION

[0015] In the following detailed description, numerous specific details are described in order to provide a thorough understanding of the invention. However, the present invention may be practiced without these specific details. In other instances, well-known methods, procedures, components and circuits have not been described in detail so as not to obscure the present invention. Further, example sizes/models/values/ranges may be given, although the present invention is not limited to these specific examples.

[0016] References in the specification to "one embodiment", "an embodiment", "an example embodiment", etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0017] An example computing device 100 shown in FIG. 1 may comprise one or more processors 110. The processors 110 may support one or more operating modes such as, for example, a real mode, a protected mode, a virtual 8086 mode, and a virtual machine mode (VMX mode). Further, the processors 110 may support one or more privilege levels or rings in each of the supported operating modes. In general, the operating modes and privilege levels of a processor 110 define the instructions available for execution and the effect of executing such instructions. More specifically, a processor 110 may be permitted to execute certain privileged instructions only if the processor 110 is in an appropriate mode and/or privilege level.

[0018] The chipset 120 may comprise one or more integrated circuit packages or chips that couple the processors 110 to memory 130, a network interface 140, a physical token 150, a virtual token 160, and other I/O devices 170 of the computing device 100 such as, for example, a mouse, keyboard, disk drive, video controller, etc. The chipset 120 may comprise a memory controller (not shown) for writing and reading data to and from the memory 130. Further, the chipset 120 and/or processors 110 may define certain regions of the memory 130 as protected memory 132 that may be accessed only by the processors 110 when in a particular operating mode (e.g. protected mode) and privilege level (e.g. 0P).

[0019] The network interface 140 generally provides a communication mechanism for the computing device 100 to communicate with one or more remote agents 190 (e.g. certification authorities, retailers, financial institutions) via a network 180. For example, the network interface 140 may comprise a 10 Mb or 100 Mb Ethernet controller, a cable modem, a digital subscriber line (DSL) modem, plain old telephone service (POTS) modem, etc. to couple the computing device 100 to the one or more remote agents 190.

[0020] In general, the physical token 150 of the computing device 100 comprises protected storage for metrics, keys and secrets and may perform various integrity functions in response to requests from the processors 110 and the chipset 120. In particular, the physical token 160 may store metrics in a trusted manner, may quote metrics in a trusted manner, may seal secrets to a particular environment (current or future), and may unseal secrets to the environment to which they were sealed.

[0021] The virtual token 160 performs in a manner similar to the physical token 150 of the computing device 100. However, the virtual token 160 may comprise more protected storage for metrics, keys and secrets since the virtual token 160 may leverage the storage capacity of the memory 130 and the protect memory 132 to store metrics, keys and secrets. Further, the virtual token 160 may perform integrity operations quicker than the physical token 150 since the virtual token 160 may leverage the processing capabilities of the processors 110 to perform such integrity operations.

[0022] As illustrated in FIG. 2, the physical token 150 may comprise one or more processing units 210 that may perform integrity functions for the computing device 100. The physical token 150 may further generate a fixed private key 220 and a corresponding fixed public key 222 in accordance with an asymmetric cryptographic algorithm such as, for example, the RSA cryptographic algorithm. In an example embodiment, the physical token 150 generates the fixed private/public key pair 220, 222 such that the fixed private key 220 and corresponding public key 222 are unique and immutable once activated.

[0023] The physical token 150 may also be affixed to or incorporated into the computing device 100 to provide some assurance to remote agents 190 that the physical token 150 is associated with only one computing device 100. For example, the physical token 150 may be incorporated into one of the chips of the chipset 120 and/or surface mounted to the main board of the computing device 100. Due to the uniqueness of the fixed physical token 150 and its incorporation into the computing device 100, a remote agent 190 may identify a computing device 100 with some certainty based upon the fixed public key 222 of the physical token 150.

[0024] Besides the fixed private/public key pair 220, 222, the physical token 150 may further generate one or more supplemental private/public key pairs 230, 232 in accordance with an asymmetric cryptographic algorithm. In an example embodiment, the computing device 100 may generate supplemental private/public key pairs 230, 232 as needed whereas the fixed private/public key pair 220, 222 is immutable. Accordingly, the computing device 100 typically provides the fixed public key 222 to only a small trusted group of entities such as, for example, a certification authority. Further, the computing device 100 typically utilizes its supplemental private/public key pairs 230, 232 for most other encryption, decryption, and digital signing operations to reduce exposure of the fixed public key 222.

[0025] The physical token 150 may further comprise one or more platform configuration registers (PCR registers) 240, 242, 244 that may be used to record and report metrics in a trusted manner. The processing units 210 may support a PCR quote operation that returns a quote or contents of an identified PCR register

240, 242, 244. The processing units 210 may also support a PCR extend operation that records a received metric in an identified PCR register 240, 242, 244. In particular, the PCR extend operation may (i) concatenate or append the received metric to an metric stored in the identified PCR register 240, 242, 244 to obtain an appended metric, (ii) hash the appended metric to obtain an updated metric that is representative of the received metric and previously metrics recorded by the identified PCR register 240, 242, 244, and (iii) store the updated metric in the PCR register 240, 242, 244.

[0026] The processing units 210 of the physical token 150 may further support a seal operation and an unseal operation. In response to a seal operation, the physical token 150 generates a sealed object comprising an object sealed to the physical token 150 and a specified device environment. Conversely, the physical token 150 may return an object of a sealed object in response to an unseal operation only if the object was sealed with a key of the physical token 150 and the current device environment satisfies environment criteria specified for the sealed object.

[0027] As used herein, the term "object" is intended to be a broad term encompassing any grouping of one or more bits regardless of structure, format, or representation. Further, the verb "hash" and related forms are used herein to refer to performing an operation upon an operand or message to produce a value or a "hash". Ideally, the hash operation generates a hash from which it is computationally infeasible to find a message with that hash and from which one cannot determine any usable information about a message with that hash. Further, the hash operation ideally generates the hash such that determining two messages which produce the same hash is computationally impossible. While the hash operation ideally has the above properties, in practice one way functions such as, for example, the Message Digest 5 function (MD5) and the Secure Hashing Algorithm 1 (SHA-1) generate hash values from which deducing the message are difficult, computationally intensive, and/or practically infeasible.

[0028] The physical token 150 may be implemented in a number of different manners. For example, the physical token 150 may be implemented to comply with the specification of the Trusted Platform Module (TPM) described in detail in the Trusted Computing Platform Alliance (TCPA) Main Specification, Version 1.1, 31 July 2001.

[0029] Still referring to FIG 2, the virtual token 160 may provide virtual or software constructs that provide functionality similar to the physical token 150. In particular, the virtual token 160 may comprise one or more virtual processing units 250 that may perform integrity functions for the computing device 100. The virtual token 160 may further generate a fixed private key 260 and a corresponding fixed public key 262 in accordance with an asymmetric cryptographic algorithm such that the fixed private key 260 and corresponding public key 262 are unique and immutable once activated.

[0030] Besides the fixed private/public key pair 260, 262, the virtual token 160 may also generate one or more supplemental private/public key pairs 270, 272 in accordance with an asymmetric cryptographic algorithm. The virtual token 160 may further comprise one or more virtual PCR registers 280 that may be used to record and report metrics in a trusted manner.

[0031] An example trusted operating environment 300 is shown in FIG. 3. The computing device 100 may utilize the operating modes and the privilege levels of the processors 110 to establish the trusted operating environment 300. As shown, the trusted operating environment 300 may comprise a trusted virtual machine kernel or monitor 310, one or more standard virtual machines (standard VMs) 320, and one or more trusted virtual machines (trusted VMs) 330. The monitor 310 of the operating environment 300 executes in the protected mode at the most privileged processor ring (e.g. 0P) to manage security and privilege barriers between the virtual machines 320, 330. Further, the monitor 310 may comprise code that implements the functionality of the virtual token 160. Alternatively, the virtual token 160 may be implemented with a separate VT software module.

[0032] The standard VM 320 may comprise an operating system 322 that executes at the most privileged processor ring of the VMX mode (e.g. 0D), and one or more applications 324 that execute at a lower privileged processor ring of the VMX mode (e.g. 3D). Since the processor ring in which the monitor 310 executes is more privileged than the processor ring in which the operating system 322 executes, the operating system 322 does not have unfettered control of the computing device 100 but instead is subject to the control and restraints of the monitor 310. In



particular, the monitor 310 may prevent the operating system 322 and its applications 324 from accessing protected memory 132 and the physical token 150.

[0033] The monitor 310 may perform one or more measurements of the trusted kernel 332 such as a hash of the kernel code to obtain one or more metrics, may cause the physical token 150 to extend an identified PCR register 244 with the metrics of the kernel 332, and may record the metrics in an associated PCR log stored in protected memory 132. Further, the monitor 310 may establish the trusted VM 330 in protected memory 132 and launch the trusted kernel 332 in the established trusted VM 330.

[0034] Similarly, the trusted kernel 332 may take one or more measurements of an applet or application 334 such as a hash of the applet code to obtain one or more metrics. The trusted kernel 332 via the monitor 310 may then cause the physical token 150 to extend an identified PCR register 244 with the metrics of the applet 334. The trusted kernel 332 may further record the metrics in an associated PCR log stored in protected memory 132. Further, the trusted kernel 332 may launch the trusted applet 334 in the established trusted VM 330 of the protected memory 132.

[0035] In response to initiating the trusted operating environment 300 of FIG. 3, the computing device 100 also records metrics for the monitor 310 and the virtual token 160 in the monitor PCR register 240 and also records metrics for the hardware in the trusted support (TS) PCR register 242. The trusted operating environment 300 may be initiated in response to various events such as, for example, system startup, an application request, an operating system request, etc.

[0036] In an example embodiment, the computing device 100 obtains and records metrics for the monitor 310 and the virtual token 160 as follows. The processor 110 hashes a monitor software module to obtain a monitor metric. The processor 110 then causes the physical token 150 to extend the monitor PCR register 240 with the monitor metric, and records the monitor metric in a monitor PCR log stored in protected memory 132. Further, if the virtual token (VT) functionality is implemented as a separate software module, the processor 110 hashes the VT software module to obtain a VT metric. The processor 110 then causes the physical token 150 to further extend the monitor PCR register 240 with the VT metric and

records the VT metric in the monitor PCR log. The monitor PCR register 240 now comprises a value that is representative of both the monitor 310 and the virtual token 160.

[0037] The computing device 100 may further record metrics for the hardware in TS PCR register 242 to provide a record of the hardware support for trusted operating environments. In an example embodiment, the processor 110 may obtain hardware identifiers such as, for example, processor family, processor version, processor microcode version, chipset version, and physical token version of the processors 110, chipset 120, and physical token 150. The processor 110 may then extend the TS PCR register 242 with obtained hardware identifiers and record the hardware identifiers in a TS PCR log stored in protected memory 132. Alternatively, the processor 110 may use more than one PCR register to record the hardware identifiers. For example, the processor 110 may store each obtained hardware identifier into a separate PCR register 240, 242, 244.

[0038] Referring now to FIG. 4, an example seal operation is illustrated. The seal operation in general results in a sealing component such as, for example, the monitor 310, the kernel 332, trusted applets 334, operating system 322, application 324, the physical token 150, and/or the virtual token 160 generating a sealed object SObj that is sealed to a token (e.g. physical token 150 or virtual token 160).

[0039] In block 410, the sealing component requests a token (e.g. the physical token 150 or the virtual token 160) to perform a seal operation. The sealing component in block 414 specifies a public token key SK (e.g. public keys 222, 232, 262, 272) to use for the seal operation. The sealing component provides the token with an object Obj to seal (block 418), and specifies environment criteria to which the object Obj is to be sealed (block 422). For example, the sealing component in block 422 may specify the environment criteria by identifying one or more PCR registers (e.g. PCR registers 240, 242, 244, 280) of the token that have metrics for one or more aspects of the current device environment. After providing the token with the above information, the sealing component in block 424 requests that the token complete the seal operation.

[0040] In response to the request to complete the seal operation, the token in block 426 generates a seal record SealRec that specifies the environment criteria (e.g. quotes of PCR registers 240, 242, 244, 280) to which the object Obj is sealed. In an example embodiment, the token generates the seal record SealRec such that it includes a unique identifier that is built-in the token. In block 430, the token hashes the object Obj and the seal record SealRec to obtain a digest value DV that attests to the integrity of the object Obj and the seal record SealRec. The token in block 434 generates a sealed object SObj by encrypting the object Obj, the seal record SealRec, and the digest value DV using an asymmetric cryptographic algorithm and the public token key SK. In block 438, the token provides the sealing component with the sealed object SObj that comprises the object Obj, the seal record SealRec, and the digest value DV encrypted with the public token key SK.

[0041] Referring now to FIG. 5, an example unseal operation is illustrated. The unseal operation in general results in an unsealing component such as, for example, the monitor 310, the kernel 332, trusted applets 334, operating system 322, application 324, the physical token 150, and/or the virtual token 160 retrieving an object Obj that has been sealed to a token (e.g. physical token 150 or virtual token 160).

[0042] In block 510, the unsealing component requests a token (e.g. the physical token 150 or the virtual token 160) to perform an unseal operation. The unsealing component in block 514 specifies a private token key UK (e.g. private keys 220, 230, 260, 270) to use for the unseal operation. The unsealing component provides the token with a sealed object SObj to be unsealed in block 518. In block 522, the unsealing component requests the token to complete the unseal operation.

[0043] The token in block 526 decrypts the sealed object SObj using the private token key UK specified by the unsealing component in block 514. If the private token key UK corresponds to the private token key SK used to seal the sealed object SObj, the token in block 526 obtains the object Obj, the seal record SealRec, and the digest value DV. Otherwise, the token encounters an error condition and/or obtains corrupted representations of the object Obj, the seal record SealRec, and the digest value DV.

[0044] In block 530, the token hashes the obtained object Obj and seal record SealRec to obtain a computed digest value CDV. In block 534, the token determines whether the obtained object Obj and seal record SealRec have been corrupted. For example, the token may determine that the object Obj, seal record SealRec, and/or digest value DV has been corrupted in response to the computed digest value CDV and the digest value DV having a predetermined relationship (e.g. not equal).

[0045] In response to determining that the object Obj, seal record SealRec, and/or digest value DV is corrupted, the token in block 538 aborts the unseal operation and erases the object Obj, the seal record SealRec, the digest value DV, and the computed digest value CDV from the token. Otherwise, the token in block 542 determines whether the current device environment satisfies the environment criteria specified for the sealed object SObj. For example, the token may determine that the current device environment satisfies the environment criteria if quotes of its PCR registers (e.g. PCR registers 240, 242, 244, 280) specified by the seal record SealRec have a predetermined relationship (e.g. equal) with the quotes of the seal record SealRec and if the unique identifier obtained from the seal record SealRec has a predetermined relation (e.g. equal) with the unique identifier built-in the token.

[0046] In response to determining that the current device environment does not satisfy the environment criteria, the token in block 538 aborts the unseal operation and erases the object Obj, the seal record SealRec, the digest value DV, and the computed digest value CDV from the token. Otherwise, the token provides the object Obj to the unsealing component in block 546.

[0047] The above example seal and unseal operations use a public key SK to seal an object Obj and a private key UK to unseal an object Obj via an asymmetric cryptographic algorithm. However, the token may alternatively use a single key SUK to both seal an object Obj and unseal an object Obj using a symmetric cryptographic algorithm. For example, the token may comprise an embedded key SUK that is used to seal and unseal objects via a symmetric cryptographic algorithm (e.g. DES, 3DES, AES, etc.).

[0048] An example multi-token seal operation is illustrated in FIG. 6. The multi-token seal operation in general results in a sealing component such as, for example, the monitor 310, the kernel 332, trusted applets 334, operating system 322, application 324, the physical token 150, and/or the virtual token 160 generating a multi-token sealed object MSObj that is sealed to two or more tokens (e.g. physical token 150 and virtual token 160). The following example multi-token seal operation is described in the context of a component sealing an object Obj to the physical token 150 and the virtual token 160 of the example computing device 100. However, it should be appreciated that the sealing component may alternatively seal the object Obj to more than two tokens and that the sealing component may seal the object Obj to different combinations of tokens (e.g. two physical tokens).

[0049] In block 610, a sealing component generates a key K using a random number generator such that key K comprises random uniqueness. It should be appreciated that the random number generator may generate pseudo-random numbers thus resulting in the key K comprising pseudo-random uniqueness. Hereinafter, the general term "random" is used to denote both truly-random and pseudo-random properties. The sealing component in block 620 encrypts the object Obj using the key K and a symmetric cryptographic algorithm to obtain an encrypted object EObj.

[0050] To obtain a sealed encrypted object SEObj, the sealing component in block 630 requests the virtual token 160 to seal the encrypted object EObj to specified environment criteria using a specified key SK of the virtual token 160. For example, the sealing component may specify the environment criteria by identifying one or more PCR registers 280 of the virtual token 160 and may request the virtual token 160 to seal the encrypted object EObj using its supplemental public key 272. In response to the request, the virtual token 160 in an example embodiment performs a seal operation in the manner described above in regard to FIG. 4. For example, the virtual token 160 in response to the seal request may generate a sealed encrypted object SEObj that comprises the encrypted object EObj, a seal record SealRec, and a digest value DV that attests to the integrity of the encrypted object EObj and the seal record SealRec, and may provide the sealing component with the sealed encrypted object SEObj.

[0051] To obtain a sealed key Seal K, the sealing component in block 640 requests the physical token 150 to seal the key K to specified environment criteria using a specified key SK of the physical token 150. For example, the sealing component may specify the environment criteria by identifying one or more PCR registers 240, 242, 244 of the physical token 150 and may request the physical token 150 seal the key K using its supplemental public key 232. In an example embodiment, the sealing component specifies at least the monitor PCR register 240 that contains a metric of the virtual token 160 in order to prevent the key K from being released to an operating environment that does not include the correct virtual token 160. In response to the request, the physical token 150 in an example embodiment performs a seal operation in the manner described above in regard to FIG. 4. For example, the physical token 150 in response to the seal operation request may generate a sealed key SealK that comprises the key K, a seal record SealRec, and a digest value DV that attests to the integrity of the key K and the seal record SealRec, and may provide the sealing component with the sealed key SealK.

[0052] As a result of requesting the above seal operations, the sealing component obtains a multi-token sealed object MSObj that comprises two sealed portions: the sealed encrypted object SEObj and the sealed key SealK. The sealing component in block 650 may store the sealed portions of the multi-token sealed object MSObj. For example, the sealing component may store the sealed encrypted object SEObj and the sealed key SealK to a machine readable medium such as, for example, memory 130, a local hard drive of the I/O devices 170 or a remote network store via the network controller 140.

[0053] An example multi-token unseal operation is illustrated in FIG. 7. The multi-token unseal operation in general results in an unsealing component such as, for example, the monitor 310, the kernel 332, trusted applets 334, operating system 322, application 324, the physical token 150, and/or the virtual token 160 retrieving an object Obj that has been sealed to two or more tokens (e.g. physical token 150 and virtual token 160). The following example multi-token unseal operation is described in the context of a component unsealing an object Obj that has been sealed to the physical token 150 and the virtual token 160 of the example computing device 100. However, it should be appreciated that the unsealing

component may alternatively unseal an object Obj that has been sealed to more than two tokens and that the unsealing component may unseal an object Obj that has been sealed to different combinations of tokens (e.g. two physical tokens).

[0054] In block 710, the unsealing component receives a sealed encrypted object SEObj and corresponding sealed key SealK. For example, the unsealing component may receive the sealed encrypted object SEObj and corresponding sealed key SealK in response to retrieving them from a machine readable medium such as, for example, memory 130, a local hard disk or a remote network store.

[0055] To obtain the encrypted object EObj, the unsealing component in block 720 provides the virtual token 160 with the sealed encrypted object SEObj and requests the virtual token 160 to unseal the sealed encrypted object SEObj. In response to the request, the virtual token 160 in an example embodiment performs an unseal operation in the manner described above in regard to FIG. 5. In particular, the virtual token 160 in an example embodiment provides the unsealing component with the encrypted object EObj only if the current device environment satisfies the environment criteria specified for the sealed encrypted object SEObj.

[0056] To obtain the key K, the unsealing component in block 730 provides the physical token 150 with the sealed key SealK and requests the physical token 150 to unseal the sealed key SealK. In response to the request, the physical token 150 in an example embodiment performs an unseal operation in the manner described above in regard to FIG. 5. In particular, the physical token 150 in an example embodiment provides the unsealing component with the key K only if the current device environment satisfies the environment criteria specified for the sealed key SealK.

[0057] The unsealing component in block 740 decrypts the encrypted object EObj using the key K and a symmetric cryptographic algorithm. As a result of decrypting the encrypted object, the unsealing component obtains the object Obj that was sealed to the physical token 150 and the virtual token 160.

[0058] The above example multi-token seal and unseal operations of FIGS. 6 and 7 use a single symmetric key K to encrypt the object Obj and decrypt the object Obj using a symmetric cryptographic algorithm. However, the multi-token seal and unseal operations may alternatively use a public key PubK to encrypt the object and a private key PrivK to decrypt the object Obj. For example, the token in block 610 may generate a private key PrivK and corresponding public key PubK and may provide the sealing component with the public key PubK. The sealing component in block 620 may encrypt the object Obj using the public key PubK. Further, the sealing component in block 640 may request the physical token 150 to seal the private key PrivK corresponding to the public key PubK, and the unsealing component may then later decrypt the encrypted object EObj using the private key PubK in block 740.

[0059] Further, a sealing component may seal the encrypted object EObj to more than two tokens. For example, the sealing component may divide the encrypted object EObj into two or more portions and may seal the portions to different virtual and/or physical tokens. The sealing component may similarly divide the key to decrypt the encrypted object EObj (e.g. the symmetric key K or the asymmetric private key PrivK) into two or more portions and seal the portions to different virtual and/or physical tokens. Alternatively, the sealing component may divide the object Obj into multiple objects Obj. The sealing object may then encrypt each Obj object with a different key K to obtain a set of encrypted objects EObj and may seal each encrypted object EObj and associated keys K to a different set of tokens.

[0060] Another example multi-token seal operation is illustrated in FIG. 8. The multi-token seal operation in general results in a sealing component such as, for example, the monitor 310, the kernel 332, trusted applets 334, operating system 322, application 324, the physical token 150, and/or the virtual token 160 generating a multi-token sealed object MSObj that is sealed to two or more tokens (e.g. physical token 150 and virtual token 160). The following example multi-token seal operation is described in the context of a sealing component sealing an object Obj to the physical token 150 and the virtual token 160 of the example computing device 100. However, it should be appreciated that the sealing component may alternatively seal the object Obj to more than two tokens and that the sealing component may seal the object Obj to different combinations of tokens (e.g. two



physical tokens).

[0061] In block 810, a sealing component generates a first key K1 and a second key K2 using a random number generator such that keys K1 and K2 each comprises random uniqueness. The sealing component in block 820 generates a third key K3 based upon the first key K1, second key K2, and a key generation algorithm. For example, the sealing component may generate the third key K3 by performing a bitwise exclusive-OR (XOR) operation of the first key K1 and the second key K2. However, the sealing component may use other key generation algorithms to obtain the third key K3 from the keys K1, K2. The sealing component in block 825 encrypts the object Obj using the third key K3 and a symmetric cryptographic algorithm to obtain an encrypted object EObj.

[0062] To obtain a sealed first key SealK1, the sealing component in block 830 requests the virtual token 160 to seal the first key K1 to specified environment criteria using a specified key SK of the virtual token 160. For example, the sealing component may specify the environment criteria by identifying one or more PCR registers 280 of the virtual token 160 and may request the virtual token 160 seal the first key K1 using its supplemental public key 272. In response to the request, the virtual token 160 in an example embodiment performs a seal operation in the manner described above in regard to FIG. 4 and provides the sealing component with a sealed first key SealK1 that comprises the first key K1, a seal record SealRec, and a digest value DV that attests to the integrity of the first key K1 and the seal record SealRec.

[0063] To obtain a sealed second key SealK2, the sealing component in block 840 requests the physical token 150 to seal the second key K2 to specified environment criteria using a specified key SK of the physical token 150. For example, the sealing component may specify the environment criteria by identifying one or more PCR registers 240, 242, 244 of the physical token 150 and may request the physical token 150 seal the second key K2 using its supplemental public key 232. In an example embodiment, the sealing component specifies at least the monitor PCR register 240 that contains a metric of the virtual token 160 in order to prevent the key K from being released to an operating environment that does not include the correct virtual token 160. In response to the request, the physical token 150 in an

example embodiment performs a seal operation in the manner described above in regard to FIG. 4 to obtain a sealed second key SealK2. The physical token 150 further provides the encryption object with the sealed second key SealK2 that comprises the second key K2, a seal record SealRec, and a digest value DV that attests to the integrity of the key K and the seal record SealRec.

[0064] The sealing component now has a multi-token sealed object MSObj that comprises three sealed portions: the encrypted object EObj, the sealed first key SealK1, and the sealed second key SealK2. The sealing component in block 850 may store the sealed portions of the multi-token sealed object MSObj. For example, the sealing component may store the encrypted object EObj, the sealed first key SealK1, and the sealed second key SealK2 to a machine readable medium such as, for example, memory 130, a local hard drive of the I/O devices 170 or a remote network store via the network controller 140.

[0065] An example multi-token unseal operation is illustrated in FIG. 9. The multi-token unseal operation in general results in an unsealing component such as, for example, the monitor 310, the kernel 332, trusted applets 334, operating system 322, application 324, the physical token 150, and/or the virtual token 160 retrieving an object Obj that has been sealed to two or more tokens (e.g. physical token 150 and virtual token 160). The following example multi-token unseal operation is described in the context of an unsealing component unsealing an object Obj that has been sealed to the physical token 150 and the virtual token 160 of the example computing device 100. However, it should be appreciated that the unsealing component may alternatively unseal an object Obj that has been sealed to more than two tokens and that the unsealing component may unseal an object Obj that has been sealed to different combinations of tokens (e.g. two physical tokens).

[0066] In block 910, the unsealing component receives a multi-token sealed object MSObj that includes an encrypted object EObj, a first sealed key SealK1, and a second sealed key SealK2. For example, the unsealing component may receive the encrypted object EObj and sealed keys SealK1, SealK2 in response to retrieving them from a machine readable medium such as, for example, memory 130, a local hard disk or a remote network store.

[0067] In block 920, the unsealing component provides the virtual token 160 with the sealed first key SealK1 and requests the virtual token 160 to unseal the sealed first key SealK1. In response to the request, the virtual token 160 in an example embodiment performs an unseal operation in the manner described above in regard to FIG. 5. In particular, the virtual token 160 in an example embodiment provides the unsealing component with the first key K1 only if the current device environment satisfies the environment criteria specified for the sealed first key SealK1.

[0068] The unsealing component in block 930 provides the physical token 150 with the sealed second key SealK2 and requests the physical token 150 to unseal the sealed second key SealK2. In response to the request, the physical token 150 in an example embodiment performs an unseal operation in the manner described above in regard to FIG. 5. In particular, the physical token 150 in an example embodiment provides the unsealing component with the second key K2 only if the current device environment satisfies the environment criteria specified for the sealed second key SealK2.

[0069] In block 940, the unsealing component generates a third key K3 based upon the first key K1, second key K2, and a key generation algorithm. For example, the unsealing component may generate the third key K3 by performing a bitwise exclusive-OR (XOR) operation of the first key K1 and the second key K2. However, the unsealing component may use other key generation algorithms to obtain the third key K3 from the keys K1, K2.

[0070] The unsealing component in block 950 decrypts the encrypted object EObj using the third key K3 and a symmetric cryptographic algorithm. As a result of decrypting the encrypted object EObj, the unsealing component obtains the object Obj that was sealed to the physical token 150 and the virtual token 160.

[0071] The above multi-token seal and unseal operations of FIGS. 8 and 9 are described in the context of sealing an object Obj to two tokens. However, the sealing component may alternatively seal the object to more than two tokens. For example, the sealing component in block 810 may generate three or more keys and may generate an encryption key in block 820 from the generated keys. The sealing

component in block 825 may encrypt the object with the encryption key generated in block 820. The sealing component may further seal the keys used to generate the encryption key to various tokens of the computing device 100.

[0072] The computing device 100 may perform all or a subset of the seal/unseal operations (FIG. 4 and 5) and/or multi-token seal/unseal operations (FIGS. 6–9) in response to executing instructions of a machine readable medium such as, for example, read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; and/or electrical, optical, acoustical or other form of propagated signals such as, for example, carrier waves, infrared signals, digital signals, analog signals. Furthermore, while FIGS. 4–9 are illustrated as a sequence of operations, the computing device 100 perform various illustrated operations of the seal/unseal operations (FIGS. 4 and 5) and/or multi-token seal/unseal operations (FIGS. 6–9) in parallel or in a different order.

[0073] While certain features of the invention have been described with reference to example embodiments, the description is not intended to be construed in a limiting sense. Various modifications of the example embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.